BDTIC www.bdtic.com/Semiconductor

ATV2500 Application Example: Video Frame Grabber

Introduction

This application note shows how the ATV2500 can be used to incorporate multiple control or logic functions into a single programmable logic device. The design example which is used is a simple NTSC video frame grabber. The ATV2500 is used to generate all of the control and addressing for the frame grabber. The application note includes a description of the frame grabber design and implementation using the ATV2500. The ABEL™ source code for the ATV2500 is included for reference and is also available from the PLD applications group on floppy disk.

ATV2500 Description

The ATV2500 is a high density programmable logic device which features 24 I/O pins and 14 input-only pins. Each I/O pin is associated with a logic macrocell (see Figures 1 and 2). The output can be configured as either combinatorial or registered. Each macrocell contains two flip-flops, 12 product terms which can be split into three separate sum terms, and an output enable. Each flip-flop has a clock term and an asynchronous reset term. Groups of four or eight flip-flops each have a common synchronous preset product term.

Each macrocell has a feedback path from the pin and from each register. This makes it possible to bury both registers and use the pin for either a combinatorial output or an input pin. A global bus routes all pins and register feedbacks to every logic cell.

Figure 1. ATV2500 Output Logic, Registered[1]

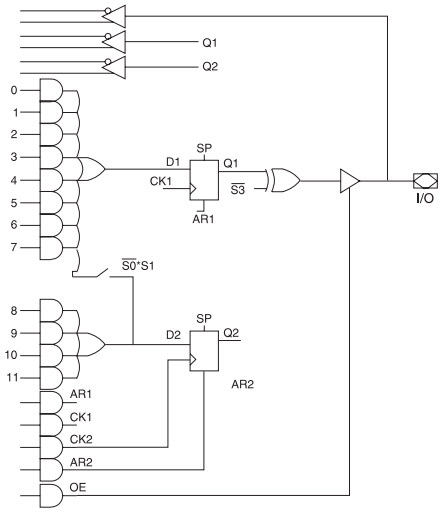
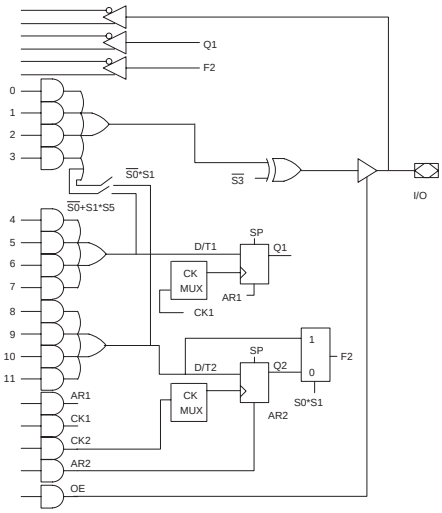


Figure 2. ATV2500 Output Logic, Combinatorial[1]



Note: 1. This diagram shows equivalent logic functions, not necessarily the actual circuit implementation.

Programmable Logic Device

Application Note

Rev. 0251D–08/99

Frame Grabber Design Considerations

The basic idea behind a frame grabber is to sample and store a frame of video data. Once the data is stored, it can be re-displayed, enhanced or saved to a file. In this example, the input video signal is converted with an A/D converter and stored in RAM. To display the buffered data, the RAM addresses are cycled and the data is converted back to a video signal through a D/A converter.

Video Basics

The NTSC (National Television Standards Committee) composite video signal is the standard used by most television and video systems in North America. The signal is composed of four components: luminance (brightness), chrominance (color), audio and synchronization. An NTSC video image or frame is composed of 525 scan lines. Each frame is actually divided into two fields of 262.5 scan lines which are interlaced. The fields start with a vertical sync period followed by the scan lines (see Figure 3). Each scan line consists of a horizontal sync period, color burst and video information (see Figure 4).

Timing

The frame grabber samples the entire video frame, including the horizontal and vertical sync periods. Then, in order to generate a video image from the stored data, the data is simply converted back to an analog signal at the same rate it was sampled.

The refresh rate for each field is 59.94 Hz (16.683 ms per field), which means the refresh rate for the whole frame is 29.97 Hz (33.366 ms per frame). The number of bits in each sample and the sampling frequency determine the resolution of the reconstructed video image. An 8-bit sample size was chosen for this example since 8-bit A/D converters are readily available. An 8-bit sample size will allow 256 levels of intensity, which is plenty for the purposes of this design. In order to generate a reasonable image, the sample frequency should be at least twice the NTSC color burst frequency of 3.579545 MHz. A sample frequency of 7.5 MHz was chosen, which is a little more than twice the color burst frequency.

The total number of samples required for each frame will be 253245 (7.5 MHz x 16.683 ms x 2).

Interface

The user interface consists of a single button. When the button is depressed, the frame grabber will pass the video signal through to the output. When the button is released, the data is sampled. The converted video signal is monitored to detect a vertical sync. Once the vertical sync is detected, 253245 samples from the video input are stored. Following the sampling, the addresses are continuously cycled creating a frozen image. The captured frame will be displayed until the button is depressed again, causing another frame of data to be sampled and stored.

It is not necessary to start sampling during the vertical sync period since 253245 samples will contain an entire frame. As the addresses are cycled, the vertical sync portion of the video signal will be generated every 16.683 ms. However, if the data were used for any purpose other than just display, it would be more convenient to have the data start at a known point in the video signal.

Figure 3. Field Timing

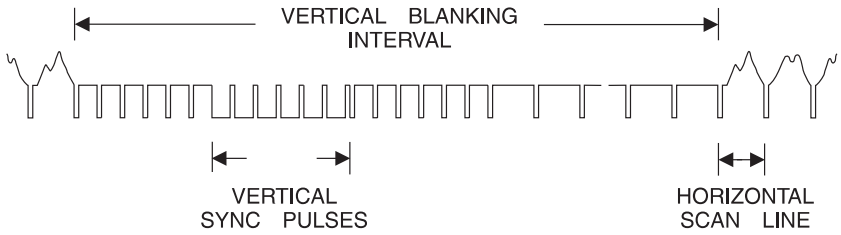
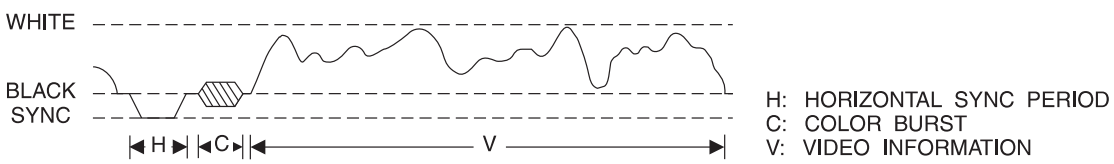


Figure 4. Horizontal Scan Line



Frame Grabber Implementation

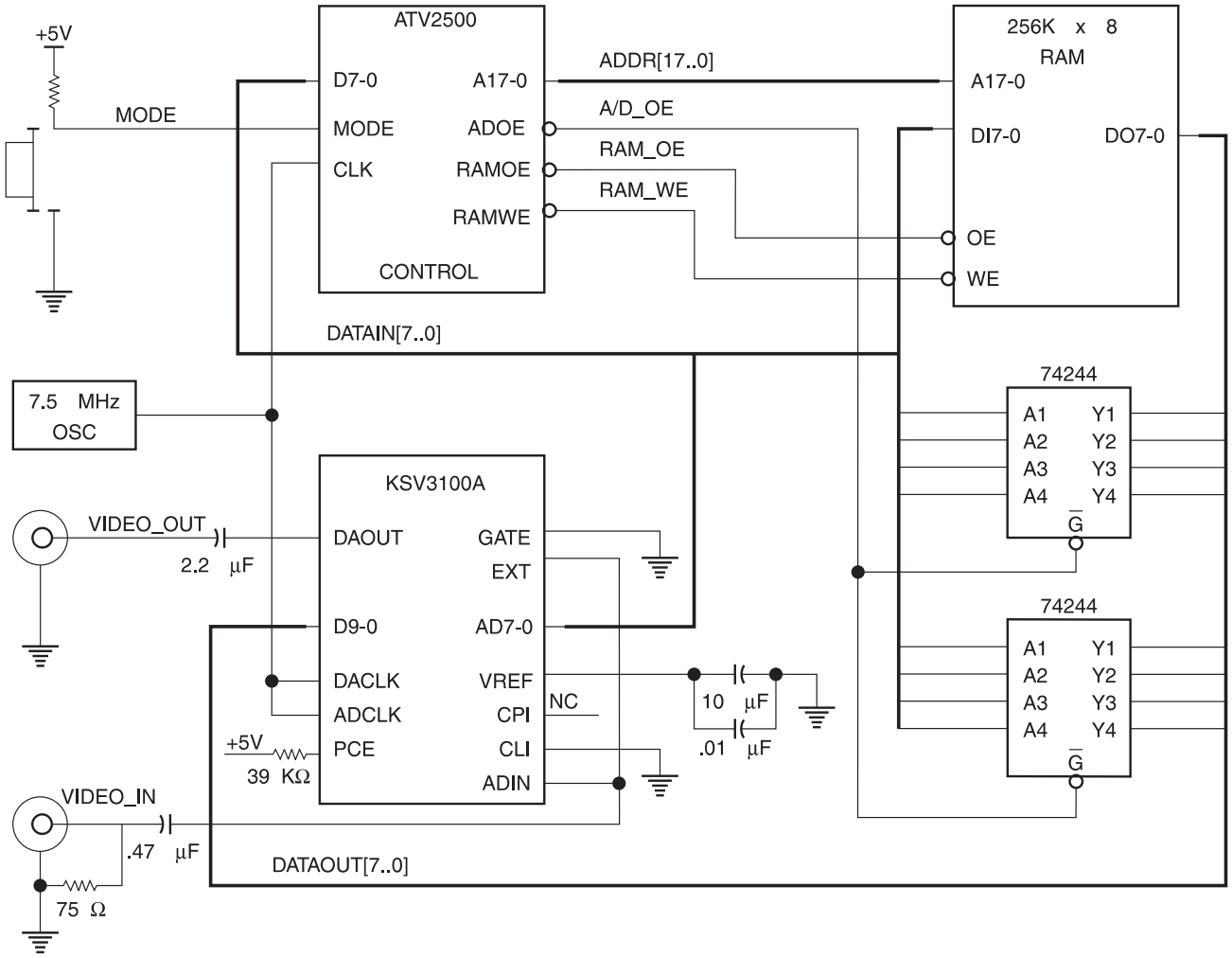
The schematic for the frame grabber is shown in Figure 5. The three basic functions are: D/A and A/D conversion, control and address generation, and storage for the sampled data.

For the A/D and D/A conversions, the Samsung KSV3100A was selected. It has both an 8-bit A/D converter and 10-bit D/A converter, along with the necessary pre-amplifier and input clamping circuit in a single device. The KSV3100A device is connected as in the recommended operating circuit in the data sheet. The 7.5 MHz system clock is used as the clock for both the A/D and D/A conversions.

The ATV2500 is used to implement all of the control and addresses for the frame grabber. The functions include a vertical sync detector, a control state machine and a RAM address counter. It receives the mode signal from the switch, the 8-bit data bus from the A/D converter and the 7.5 MHz system clock signal. It generates the bus control signals and 18-bit address for the RAM.

A 256K x 8 static RAM module is used for the frame buffer memory. A write enable signal from the controller allows the converted data to be stored in the RAM during sampling. An output enable signal is used to enable/disable the RAM onto the data bus. External buffers are used to enable the data from the output of the A/D back into the D/A when the video signal is passed through or the data is being sampled.

Figure 5. Frame Grabber Schematic



ATV2500 Control and Address Functions

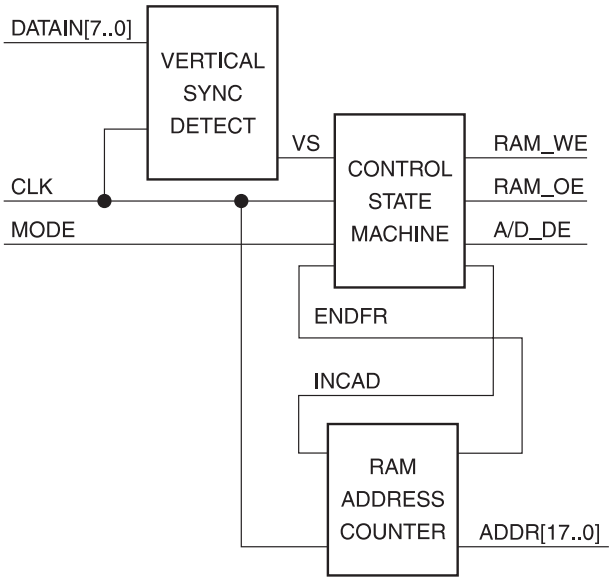
The ATV2500 is used to implement the control and address functions. These include a vertical sync detector, a control state machine and the RAM address counter. Figure 6 shows a block diagram of the ATV2500 functions.

Vertical Sync Detector

During the horizontal and vertical sync periods, the signal drops to a sync level which is lower than any other portion of the signal. This level will be clamped in the A/D converter and will become the zero value when it is converted. The horizontal sync pulse is about 4.7 µs, which will correspond to about 35 samples. The entire vertical sync period is 20 times the horizontal scan width. Within that period, there are pulses of around 31 ms when the signal is at the sync level. Each of these pulses will correspond to about 238 samples.

The vertical sync detector is a 7-bit counter which will count every sample with a zero value. If it counts 128 such samples, then it has detected one of the pulses in the vertical sync period. The counter is divided into two parts, so that the maximum number of product terms required for any count bit is four. A carry bit from the first stage is used to enable the second stage. The counter will increment whenever the input data is zero. A non-zero value will cause the counter to reset. When the count reaches 127, the VS signal is asserted.

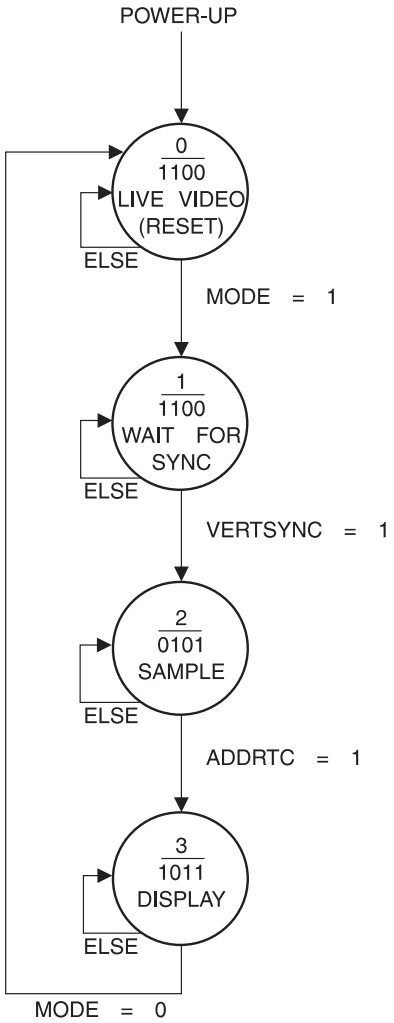
Figure 6. ATV2500 Function Block Diagram



State Machine

A state machine generates internal control for the address generation and external control signals for the RAM. The state diagram is shown in Figure 7. When the MODE input changes, indicating that the frame grabber should sample a new frame, the state machine waits for the VS signal. After the vertical sync is detected, it sets up the control signals to write the sampled data for the entire frame. When the address counter reaches the correct number of samples, the state machine changes the control signals to read the sampled data. The addresses are cycled so that the sampled frame is continuously displayed.

Figure 7. Controller State Diagram



Address Counter

The address counter is controlled by the INCAD control signal from the state machine. If the INCAD signal is asserted, the address will increment until the counter reaches the final address for the correct number of samples. When the INCAD signal is not asserted, or the final address is reached, the counter resets to 0. The 18-bit address counter is broken into two 9-bit stages, so that the maximum number of product terms required for each count bit is less than 12. A carry from the first stage is used to enable to second stage.

Resource Allocation

The address counters use registered I/O pins in the ATV2500. The MSBs of each stage of the counter have more than eight product terms, so they require an entire macrocell. The LSBs of each stage use less than eight product terms, so a buried register is available in each of those macrocells (see Figure 1). The control bits generated by the state machine use combinatorial outputs with less than four product terms, so two buried registers are available in each of those macrocells (see Figure 2). The state bits and vertical sync detect counter bits are assigned to available buried registers.

Assignment of the I/O pins is only dependent on the board layout, since there are no signal routing limitations in the ATV2500. Buried logic can be assigned to any available resources. Table 1 shows a worksheet used to allocate the logic functions in this application to the ATV2500 resources. The shaded boxes indicate resources which are used by another signal generated in the same macrocell. The empty boxes indicate available resources. In this design, four input pins, one I/O pin and associated macrocell plus 13 additional buried registers are available for expansion or design changes.

Notes on Test Vectors

In order to test that the address counter rolls over at the correct count, more than 253245 test vectors would be required, which far exceeds the number of vectors allowed in ABEL. There are a few ways to get around this problem.

The first method is to use the register preload function in PLASIM™. The counter can be preloaded with a count near to the maximum count, and then allowed to roll over. However, since JEDSIM™ does not support the preload function the JEDEC file is not checked.

Another method is to change the terminal count value to a smaller value, so that the counter would reach the terminal count in fewer vectors. Since the equations are altered, this method could only be used for ABEL™ simulation.

A third method is to use the register synchronous presets to load the counter. This is the method that was used in this design. The registers which use the preset are allocated to the same ATV2500 synchronous preset groups. A spare input pin (PRELD) is used to control the preset in the test vectors. Using this method, the vectors can also be used on the programmer.

Summary

In this example, the address and control functions are greatly simplified by using a single complex programmable logic device. The ATV2500 is ideal for this application since it has enough inputs to accept the control and data signals, and the necessary I/O's to generate the RAM addresses and control. The ability to combine or separate the sum terms in each macrocell allows for maximum usage of the available resources. The functions which require large numbers of product terms do not have to be split into smaller pieces. The functions which require fewer numbers of product terms do not waste an entire macrocell. The left-over buried logic in those macrocells can be used for the control state machine and the vertical sync detector.

Table 1. ATV2500 ABEL Pin/Node Assignment Worksheet Example

Input	Signal	Input	Signal
1	CLK	21	D4
2	MODE	22	D5
3		23	D6
17	D0	37	D7
18	D1	38	
19	D2	39	
20	D3	40	(PRELD)

I/O	Signal	Q2	Signal	Q1	Signal
4	A0	41	SYNC0	217	
5	A1	42	SYNC1	218	
6	A2	43	SYNC2	219	
7	A3	44	SYNC3	220	
8	A4	45	SYNCARRY	221	
9	A5	46	SYNC4	222	
11	A8	47		223	
12	A10	48		224	
13	A6	49	SYNC5	225	
14	A7	50	SYNC6	226	
15	A9	51	ENDFR	227	
16	A13	52	ADCARRY	228	
24	A11	53		229	
25	A12	54		230	
26	A14	55		231	
27	A15	56		232	
28	A16	57		233	
29	A17	58		234	
31	VS	59	ST0	235	
32	RAMWE	60	ST1	236	
33	RAMOE	61		237	
34	ADOE	62		238	
35	INCAD	63		239	
36		64		240	

ABEL Source Code

```
module CONTROL ;
title 'Application example for the ATV2500 - Frame Grabber Controller
Atmel Corporation PLD - (408)436-4333 PLD Applications Hotline
Wendey Mueller - January 10, 1992'


CONTROL device 'P2500';


CLK,MODE            pin 1,2;                                    "system clock and mode signal from button
D0,D1,D2,D3         pin 17,18,19,20;                            "digitized video signal from A/D
D4,D5,D6,D7         pin 21,22,23,37;


A0,A1,A2,A3,A4      pin 4,5,6,7,8 istype 'reg,buffer';          "RAM address outputs
A5,A6,A7,A8,A9      pin 9,13,14,11,15 istype 'reg,buffer';
A10,A11,A12,A13,A14 pin 12,24,25,16,26 istype 'reg,buffer';
A15,A16,A17         pin 27,28,29 istype 'reg,buffer';
RAMWE,RAMOE         pin 32,33 istype 'com,buffer';              "RAM control signals
ADOE,INCAD          pin 34,35 istype 'com,buffer';              "address counter control
VS                  pin 31 istype 'com,buffer';                 "vertical sync detected
ADCARRY,SYNCARRY    node 52,45 istype 'reg,buffer';             "counter carry signals
ENDFR               node 51 istype 'reg,buffer';                "end of frame detected
ST0,ST1             node 59,60 istype 'reg,buffer';             "state bits
SYNC0,SYNC1,SYNC2   node 41,42,43 istype 'reg,buffer';
SYNC3,SYNC4,SYNC5   node 44,46,49 istype 'reg,buffer';          "vertical sync detect
SYNC6               node 50 istype 'reg,buffer';                "counter

"Use spare input pin as preset signal for testing counter


PRELD               pin 3;


C,X,Z,H,L,P = .C.,.X.,.Z.,1,0,.P.;


"Create buses


DATA   = [D7..D0];          "data bus
ADDR   = [A17..A0];         "address counter
ADDRA  = [A8..A0];          "address counter LSB
ADDRB  = [A17..A9];         "address counter MSB
SYNC   = [SYNC6..SYNC0];    "vertical sync detect counter
SYNCA  = [SYNC3..SYNC0];    "vertical sync detect counter LSB
SYNCB  = [SYNC6..SYNC4];    "vertical sync detect counter MSB
STMACH = [ST1,ST0];         "state bits


"Define states


S0     = [0,0];
S1     = [0,1];
S2     = [1,0];
S3     = [1,1];


EQUATIONS


"256K address counter - increment when INCAD is true and address
"                       has not reached end of frame, otherwise reset


ADDRA.D = ((ADDRA.FB + 1) & INCAD & !ENDFR); "LSB stage
```

```
ADDRA.CK = CLK;

ADCARRY.D = (ADDRA.FB == 510); “synchronous carry bit from 1st stage
ADCARRY.CK = CLK;

ADDRB.D = ((ADDRB.FB + 1) & INCAD & !ENDFR & ADCARRY) “MSB stage
         # (ADDRB.FB & INCAD & !ENDFR & !ADCARRY);
ADDRB.CK = CLK;

ENDFR.D = (ADDR.FB == 253243); “set ENDFR when address reaches 253243
ENDFR.CK = CLK;

“Use synchronous preset to simplify testing the address counter by
“preloading it with a value close to the last address.

A17.SP = PRELD;
A16.SP = PRELD;
A15.SP = PRELD;
A14.SP = PRELD;
A12.SP = PRELD;
A11.SP = PRELD;
A10.SP = PRELD;
A8.SP  = PRELD;

“Vertical sync detect counter - increment if data is 0, otherwise reset

SYNCA.D = (SYNCA.FB + 1) & (DATA == 0); “LSB stage
SYNCA.CK = CLK;

SYNCARRY.D = (SYNCA.FB == 14); “synchronous carry bit from 1st stage
SYNCARRY.CK = CLK;

SYNCB.D = (SYNCB.FB + 1) & (DATA == 0) & SYNCARRY  “MSB stage
         # SYNCB.FB & (DATA == 0) & !SYNCARRY;
SYNCB.CK = CLK;

VS = (SYNC.FB == 127); “set vertical sync detected bit if count reaches 128

STMACH.CK = CLK;

“State Machine Controller -
“
“Inputs: MODE,VS,ENDFR
“Outputs: RAMWE,RAMOE,ADOE,INCAD

STATE_DIAGRAM STMACH

STATE S0:                   “Reset, live video
    RAMWE = 1;
    RAMOE = 1;
    ADOE = 0;
    INCAD = 0;
    IF (MODE) THEN S1       “If capture mode, go to state 1
    ELSE S0                 “else wait for mode change
```

```
STATE S1:                       “Wait for vertical sync signal
    RAMWE = 1;
    RAMOE = 1;
    ADOE = 0;
    INCAD = 0;
    IF (VS) THEN S2             “Vertical sync detected, start sampling data
    ELSE S1                     “else wait for vertical sync

STATE S2:                       “Sample video data
    RAMWE = 0;
    RAMOE = 1;
    ADOE = 0;
    INCAD = 1;
    IF (ENDFR) THEN S3          “Frame capture complete
    ELSE S2                     “else continue sampling

STATE S3:                       “Display frame data
    RAMWE = 1;
    RAMOE = 0;
    ADOE = 1;
    INCAD = 1;
    IF (!MODE) THEN S0          “Reset, display live video
    ELSE S3                     “else continue to display frame data

@@RADIX 16;
@@CONST ACNT = 1;
@@CONST SCNT = 0;

TEST_VECTORS (
[CLK,MODE,DATA,PRELD] - [SYNC,VS,STMACH,RAMWE,RAMOE,ADOE,INCAD,ADDR, ENDFR])

“check that the vertical sync detector resets if data is not 0 and set mode
“to start sample and display sequence
[ 0,  0,  0,   0    ] - [00,  0, 0,     1,    1,    0,   0,    00000,0    ];
[ C,  0,  0,   0    ] - [01,  0, 0,     1,    1,    0,   0,    00000,0    ];
[ C,  0,  1,   0    ] - [00,  0, 0,     1,    1,    0,   0,    00000,0    ];
[ C,  1,  1,   0    ] - [00,  0, 1,     1,    1,    0,   0,    00000,0    ];

“simulate vertical sync
@@REPEAT 7E {
@@CONST SCNT = SCNT + 1;
[ C,  1,  0,   0    ] - [SCNT,0, 1,     1,    1,    0,   0,    00000,0    ]; }
[ C,  1,  0,   0    ] - [7F,  1, 1,     1,    1,    0,   0,    00000,0    ];
[ C,  1,  0,   0    ] - [00,  0, 2,     0,    1,    0,   1,    00000,0    ];
[ C,  1,  0,   0    ] - [01,  0, 2,     0,    1,    0,   1,    00001,0    ];

“state machine has detected vertical sync and started sampling.
@@REPEAT 0FE {
@@CONST ACNT = ACNT + 1;
[ C,  1,  1,   0    ] - [00,  0, 2,     0,    1,    0,   1,    ACNT, 0    ];}
[ C,  1,  1,   0    ] - [00,  0, 2,     0,    1,    0,   1,    00100,0    ];
[ C,  1,  1,   0    ] - [00,  0, 2,     0,    1,    0,   1,    00101,0    ];

“Use preset to preload a count value near the largest address value.
[ C,  1,  1,   1    ] - [00,  0, 2,     0,    1,    0,   1,    3DD02,0    ];
[ C,  1,  1,   0    ] - [00,  0, 2,     0,    1,    0,   1,    3DD03,0    ];

“Allow counter to reach the largest address and roll over. State machine
“then changes controls to display sampled data.
```

```
@@CONST ACNT = 3DD03;
@@REPEAT 037 {
@@CONST ACNT = ACNT + 1;
[ C,  1,  1,   0    ] - [00,  0, 2,     0,    1,    0,   1,    ACNT, 0    ];}
[ C,  1,  1,   0    ] - [00,  0, 2,     0,    1,    0,   1,    3DD3B,0    ];
[ C,  1,  1,   0    ] - [00,  0, 2,     0,    1,    0,   1,    3DD3C,1    ];
[ C,  1,  1,   0    ] - [00,  0, 3,     1,    0,    1,   1,    00000,0    ];
[ C,  1,  1,   0    ] - [00,  0, 3,     1,    0,    1,   1,    00001,0    ];
[ C,  1,  1,   0    ] - [00,  0, 3,     1,    0,    1,   1,    00002,0    ];

END ;
```

BDTIC www.bdtic.com/Semiconductor